Introduction to Scala

Practical Exercises

Object Oriented Scala

1. Consider the following class, designed to represent a Person:

```
class Person ( val firstName: String,
               val lastName: String,
               val age: Int ) {
  …
}
```

   This class is not particularly well suited to immutability. In particular the age property needs to be var, or else we will be unable to model the fact that a Person gets older. Investigate how we may be able to add immutability to the class, while preserving this functionality. Don't worry about name changes at the moment.

2. Based on the question from the previous exercise, define a class User, to represent a Unix user. Using the ability to extract properties from the password file string, define a factory method that allows instances of the class to be constructed from a password file entry.

3. If the "shell" entry is something other than a normal interactive shell, it indicates that the user is not allowed to fully log in to the system. Interactive shell programs normally have a name that ends with the characters "sh". Add a method to your User class that indicates whether the user is allowed to login.

4. Following from the example in the notes illustrating the Complex type, implement a Fraction type in Scala. Include appropriate constructors, with default values where appropriate. Implement the four basic arithmetic operators +, -, *, /. Include appropriate equality test operators as well: == and !=. Ensure that the objects of this type are proper fractions – the denominator cannot be 0.

5. Enhance the Fraction class you built earlier, by adding a trait that supports ordering of the type.

6. With regard to the Unix User type from the previous exercises, some of our Unix users are to have additional privileges in the system (not the normal Unix/Linux "root" privileges, but somewhere between this and the basic privileges). Define a trait called "Privileged", and then mix this in with the User type to create a type called PriviligedUser.